Software for a New Modified Cholesky Factorization

Elizabeth Eskow and Robert B. Schnabel

CU-CS-443-89 August 1989

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

2

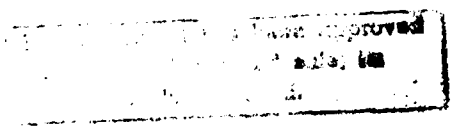
Software for a New Modified Cholesky Factorization

Elizabeth Eskow and Robert B. Schnabel

CU-CS-443-89 August 1989

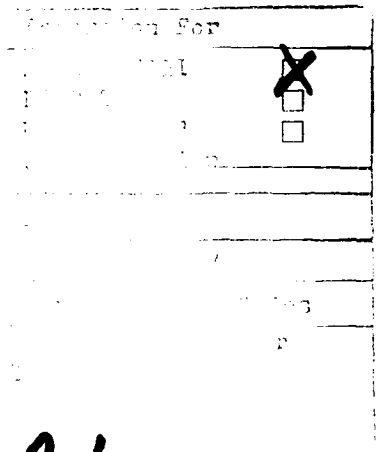
Department of Computer Science
Campus Box 430
University of Colorado,
Boulder, Colorado, 80309 USA

This research was supported by ARO grant DAAL-03-88-0086, NSF grant CCR-8702403, and NSF cooperative agreement DCR-8420944.



Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.



A-1

Abstract

This paper describes the software for a new modified Cholesky factorization recently proposed by the authors. Given a symmetric but not necessarily positive definite matrix A, the modified Cholesky factorization computes a Cholesky factorization of $A + E$, where $E = 0$ if A is safely positive definite, and E is a diagonal matrix chosen to make $A + E$ positive definite otherwise. The modified Cholesky factorization was introduced by Gill and Murray and refined by Gill, Murray and Wright, and is commonly used in optimization algorithms. Our version, which is based upon new techniques, has a considerably smaller a priori upper bound on the size of E than the Gill, Murray and Wright factorization, and appears to generally produce a smaller E, and a well-conditioned $A + E$, in practice. Its cost, like the Gill, Murray and Wright version, is only a small multiple of n^2 operations greater than the standard Cholesky factorization. Thus it may be useful in optimization algorithms. We summarize our algorithm and describe the code and its use.

1 Introduction

This paper describes the software for a new modified Cholesky factorization algorithm presented in Schnabel and Eskow [1988]. The modified Cholesky factorization is intended for matrices that are symmetric but not necessarily positive definite. Given a matrix $A \in R^{n \times n}$, the modified Cholesky factorization computes

$$P^T(A + E)P = LL^T (or LDL^T),$$

where P is a permutation matrix, and $E \in R^{n \times n}$ is 0 if A is safely positive definite, otherwise E is a nonnegative diagonal matrix chosen such that $A + E$ is safely positive definite. This type of factorization was introduced by Gill and Murray [1974] and subsequently refined by Gill, Murray, and Wright [1981]. It has become important in solving problems in optimization, and is used in many line search methods for unconstrained and constrained optimization problems (Gill, Murray, and Wright [1981]) as well as in some trust region methods (Dennis and Schnabel [1983]).

The modified Cholesky factorization of Schnabel and Eskow has superior theoretical properties to the method of Gill, Murray, and Wright, and appears to have computational advantages as well. The upper bound on $||E||_\infty$ for Schnabel and Eskow's method is at worst about $2n\alpha$, where $\alpha = \max_{i \leq i,j \leq n} |A_{ij}|$, as opposed to roughly $n^2\alpha$ for the method of Gill, Murray, and Wright. In addition, extensive computational testing of the software described in this paper was performed by Schnabel and Eskow [1988], on a set of randomly generated indefinite matrices with $n = 25$, 50 and 75. The norm of the matrix E and condition number of the $(A + E)$ computed by this algorithm were compared to those produced by the Gill, Murray, and Wright [1981] factorization. In almost all cases, $||E||_\infty$ for the new algorithm was significantly smaller, while both methods consistently produced acceptably conditioned matrices.In addition, the $||E||_\infty$ produced by the Schnabel and Eskow algorithm almost always was within a factor of 2 of the magnitude of the most negative eigenvalue of A, and often much closer. Another nice property of the new factorization (Gould [1989]) is that in applications like multifrontal methods, where A is large and sparse and no pivoting is performed, the matrix A may be generated and processed incrementally, as the factorization proceeds, whereas the Gill, Murray, and Wright method requires the entire matrix to be known, and processed, in its initialization stage. Thus, the new factorization may be useful in optimization and other contexts.

In Section 2, we give a brief description of the algorithm. Section 3 demonstrates the use of the algorithm on a small (4 x 4) example matrix and section 4 explains the parameters and organization of

the code. Appendices A and B provide a sample driver and its output, respectively, and Appendix C contains the code for the modified Cholesky factorization.

2 The Modified Cholesky Factorization Algorithm

This section briefly describes the modified Cholesky factorization algorithm presented in Schnabel and Eskow[1988]. Some detail concerning the techniques used to prevent an ill-conditioned result is included because the user has the option of adjusting two tolerances related to this. The cost of the factorization is also described. For a more detailed explanation of the entire algorithm, see the original paper.

The new modified Cholesky factorization uses a two phase approach to compute $P^T(A+E)P = LL^T$, where $A \in R^{n \times n}$ is symmetric, E is a nonnegative diagonal matrix or 0, L is a lower triangular matrix and P is a permutation matrix. Phase 1 computes the standard Cholesky factorization, and ends when the next iteration of the standard factorization would cause some diagonal element in the remaining submatrix to become non-positive. It performs diagonal pivoting based on the maximum diagonal element of the submatrix remaining to be factored. Schnabel and Eskow show that at the end of this phase, no element in the submatrix that remains to be factored is larger, in magnitude, than the sum of the magnitudes of the largest diagonal and off-diagonal elements in that submatrix originally.

Phase 2 does a modified Cholesky factorization, meaning that at each iteration, an amount $E_{jj} \geq 0$ is added to the pivot A_{jj} before the elimination step. A diagonal pivoting strategy is also used, but in this phase the pivot row is chosen to be the one with the maximum lower Gerschgorin bound estimate (see below). The amount E_{jj} added to the pivot element at each iteration is determined from the actual lower Gerschgorin bound of the pivot row. Schnabel and Eskow show that due to the choice of E_{jj}, the Gerschgorin bounds of the next remaining submatrix (after elimination) do not grow. In turn, this implies that $\|E\|_\infty$ is bounded above by the magnitude of the most negative lower Gerschgorin bound of the submatrix that remained to be factored at the start of phase 2. This, combined with the growth bound for phase 1 leads to the theoretical result mentioned in Section 1.

The entire algorithm is outlined in Algorithm 2.1 below.

Algorithm 2.1 - Modified Cholesky Decomposition

Given $A \in R^{n \times n}$ symmetric and τ_1 and τ_2 (e.g. $\tau_1 = \tau_2 = macheps^{\frac{1}{3}}$).
find factorization LL^T of $A + E$, $E \geq 0$

$\gamma := \max_{1 \leq i \leq n} |A_{ii}|$; $j := 1$
(* Phase One, A potentially positive definite *)
While $j \leq n$ do
Pivot on maximum diagonal of remaining submatrix
If $\min_{j+1 \leq i \leq n} \{A_{ii} - \frac{A_{ij}^2}{A_{jj}}\} < \tau_1 \gamma$
then go to Phase Two
else perform j^{th} iteration of standard Cholesky factorization
and increment j
(* Phase Two, A not positive definite *)
$k := j - 1$ (* k = number of iterations performed in Phase One *)
Calculate lower Gerschgorin bounds of A_{k+1}
For $j := k + 1$ to $n - 2$ do
Pivot on maximum lower Gerschgorin bound estimate
Calculate E_{jj} and add E_{jj} to A_{jj}
(* $E_{jj} = \max\{0, -A_{jj} + \max\{\sum_{i=j+1}^{n} |A_{ij}|, \tau_2\gamma\}, E_{j-1,j-1}\}$ *)
update Gerschgorin bound estimates
perform j^{th} iteration of Cholesky factorization
complete factorization of final 2 x 2 submatrix using its eigenvalues

In order to prevent $A + E$ from being singular or very ill-conditioned, the algorithm includes the following details. Let γ be the maximum diagonal element of A, and τ_1 and τ_2 be some small constants (our default choice is $\tau_1 = \tau_2 = macheps^{\frac{1}{3}}$). The switch to phase 2 is made when some diagonal element in the remaining submatrix would become less than $\tau_1\gamma$, implying that only an indefinite matrix or a positive definite matrix with condition number greater than $\frac{1}{\tau_1}$ may be perturbed. During phase 2, the the amount E_{jj} to add to A_{jj} is

$$E_{jj} = \max\{0, -A_{jj} + \max\{\sum_{i=j+1}^{n} |A_{ij}|, \tau_2\gamma\}, E_{j-1,j-1}\}$$

where $E_{j-1,j-1}$ is the amount added to $A_{j-1,j-1}$ in the previous iteration. The $\tau_2\gamma$ term in the above computation allows the condition number of $A + E$ to be bounded above. The final place in the algorithm

where the conditioning of the resultant matrix is addressed is in the final $((n-1)^{st})$ iteration of the factorization. Eigenvalues λ_{lo} and λ_{hi} of the remaining 2×2 submatrix are computed, which are then used to calculate

$$E_{n-1,n-1} = E_{n,n} = \max\{0, E_{n-2,n-2}, -\lambda_{lo} + \tau_2 * \max\{\frac{1}{1-\tau}(\lambda_{hi} - \lambda_{lo}), \gamma\}\}.$$

This causes the l_2 norm of the resultant final 2×2 submatrix to be no greater than $\frac{1}{\tau_2}$, and in practice usually results in $E_{n-1,n-1}$ having a smaller value than would otherwise be obtained using Gerschgorin bounds. The analysis in Schnabel and Eskow [1988] includes these details. In practice, the condition number of $A+E$ is usually no greater than $10/\min\{\tau_1, \tau_2\}$, although this bound does not hold in theory

Phase 2 of the algorithm pivots on an estimate of the lower Gerschgorin bounds of the remaining submatrix. Let $G_i, (j \leq i \leq n)$ denote the lower Gerschgorin bound estimates used during the j^{th} iteration. The actual lower Gerschgorin Circle Theorem bounds are computed once at the start of phase 2, giving

$$G_i = A_{ii} - \sum_{k=j}^{i-1} |A_{ik}| - \sum_{k=i+1}^{n} |A_{ki}|, \quad i = j, \cdots, n,$$

where j is the iteration in which the algorithm switches to phase 2. Thereafter, at each iteration j the bounds are estimated by

$$G_i = G_i + |A_{ij}| - \frac{|A_{ij}| \sum_{i=j+1}^{n} |A_{ij}|}{A_{jj}}, \quad i = j, \cdots, n,.$$

Since in the calculation of the estimates of the bounds, the sum $\sum_{i=j+1}^{n} |A_{ij}|$ needs only to be computed once at each iteration, the cost of computing the bound estimates is at most $n^2/2$ each additions and multiplications over the entire algorithm, whereas it would be $O(n^3)$ if the actual Gerschgorin bounds were used. The cost of the entire modified Cholesky factorization is at most $2n^2$ additions and $n^2/2$ multiplications greater than the $n^3/6+O(n)$ each multiplications and additions for the standard Cholesky factorization of positive definite matrices. If A is safely positive definite, there is no extra cost.

3 Example using the Modified Cholesky Factorization

The following discussion shows how the modified Cholesky factorization works on an example matrix of size $n = 4$. Consider the matrix

$$A = \begin{bmatrix} 0.3571 & -0.1030 & 0.0274 & -0.0459 \\ -0.1030 & 0.2525 & 0.0736 & -0.3845 \\ 0.0274 & 0.0736 & 0.2340 & -0.2878 \\ -0.0459 & -0.3845 & -0.2878 & 0.5549 \end{bmatrix}$$

The eigenvalues of this matrix are -.0767, .1442 , .4004, and .9307 and the maximum diagonal element, γ, is .5549. Let $\tau_1 = \tau_2 = 6.0555e-06$, which is the value of $macheps^{\frac{1}{3}}$ on a Sun 3/75, using double precision.

In the first iteration, A_{44} is the maximum diagonal element, therefore row and column 4 are interchanged with row and column 1. In performing the test of whether or not the $\min_{j+1 \le i \le n} \{A_{ii} - \frac{A_{ij}^2}{A_{jj}}\} < \tau_1\gamma$, the minimum occurs at $A_{22} - \frac{A_{21}^2}{A_{11}}$, which is < 0, and consequently the algorithm switches to phase 2.

The actual lower Gerschgorin bounds for the start of phase 2 are [-.1633, -.3086, -.1548, .1808]. The maximum bound is the bound for row 4, hence row and column 1 are again interchanged with row and column 4, and because this bound is greater than 0, $E_{11} = 0$.

Prior to the start of iteration 2, the updated lower Gerschgorin bound estimates become $[-.2564, -.1410, -.1401]$ for rows 2 through 4. The maximum of these is the estimate for row 4, resulting in a diagonal pivot of rows and columns 2 and 4. The actual lower Gerschgorin bound for row 2 is -.1330, therefore $E_{22} = .1330$.

In the final iteration, the eigenvalues of the remaining 2×2 submatrix are .156329 and -.052115, or λ_{lo} and λ_{hi}, respectively. The value $-\lambda_{lo} + \frac{\tau_2}{1-\tau_2}(\lambda_{hi} - \lambda_{lo})$ is .052119, which is less than E_{22}, therefore the algorithm sets both E_{33} and E_{44} to the value added to A_{22} in the previous iteration, or .1330.

The Cholesky factors and pivot vector are

$$L = \begin{bmatrix} 0.5976 & & & \\ -0.0769 & 0.8259 & & \\ 0.0458 & -0.3442 & 0.4964 & \\ -0.1724 & -0.4816 & -0.1697 & 0.3082 \end{bmatrix}, \quad E = \begin{bmatrix} 0.0 & & & \\ & 0.1330 & & \\ & & 0.1330 & \\ & & & 0.1330 \end{bmatrix} \text{ and } \overline{P} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ 2 \end{bmatrix},$$

where $P^T A P + E = LL^T$, and P is I permuted by the transformations recorded in $\overline{P}$.

The ratio $\|E\|_\infty / -\lambda_1(A)$, where $\lambda_1(A)$ is the most negative eigenvalue of A, is 1.73, and the condition number of $(A+E)$ is 21.8. In comparison, for the Gill, Murray, and Wright [1981] algorithm, $\|E\|_\infty / -\lambda_1(A)$ is 0.48, and the condition number of $(A+E)$ is 39.2.

4 Software for the Modified Cholesky Factorization

The code for the modified Cholesky factorization is a straightforward implementation of the algorithm detailed in Appendix 1 of Schnabel and Eskow [1988]. It is organized into one main user-callable subprogram containing three smaller subroutines, each of which are called only once. These three smaller subroutines serve to initialize variables at the start of the algorithm, initialize the actual Gerschgorin bounds at the start of phase 2, and compute the factorization of the final 2×2 submatrix in phase 2. The remainder of the factorization is performed by the main subroutine. In particular, while the code for pivoting in phase 1 and phase 2 is similar, it has been left in-line to prevent the necessity of having $O(n)$ function calls. Because the row and column pivoting must affect only the lower triangle of the input matrix, this code is lengthy in comparison to the remainder of the algorithm. All non-integer variables in the code are double precision.

The main subprogram is called by *modcholesky(ndim,n,A,g,macheps,τ_1,τ_2,pivot,E)* . The input parameters to this subroutine are:

- *ndim* is the dimension of matrix that contains A in the calling program.
- n is the dimension of the input matrix A.
- A is an $n \times n$ symmetric matrix (only the lower triangular portion of A, including the diagonal, is used, and it is overwritten by L).
- g is an n dimensional work vector.
- *macheps* is machine epsilon.
- τ_1 is the reciprocal of the tolerance used for determining when to switch to phase 2, i.e. $1/\tau_1$ is the minimum condition number of a positive definite input matrix which may be perturbed by the algorithm.
- τ_2 is the tolerance used for determining the maximum condition number of the final 2×2 submatrix and in the equation for E_{jj}.

The output parameters are

- L is stored in the matrix A (in the lower triangular portion, including the main diagonal.
- *pivot* is a record of how the rows and columns of the matrix were permuted during the factorization. That is, each P_i is initialized to i, and at each iteration, if rows and columns i and j are switched, then P_i and P_j are swapped.
- E is an n-vector, whose i^{th} element is the amount added to the diagonal of A at the i^{th} iteration of the factorization.

A simpler driver, called by *modchol(ndim,n,A,G,macheps,pivot,E)* is also available. This driver sets the parameters τ_1 and τ_2 to $macheps^{\frac{1}{3}}$, and the remaining input and output parameters are identical to those for *modcholesky*.

A sample driver program, *choldriver.f*, and its output are included with the code for the modified Cholesky factorization. The driver calls a function *macheps* to compute machine epsilon for double precision arithmetic. It can not be guaranteed that this function will return the correct value of *macheps* on every computer, so the user may want to check this and, if necessary, replace the call to *macheps* with a statement that assigns the actual value of machine epsilon for that computer to *eps*. The driver program also calls a separate subprogram, *mkmatrix.f*, to generate random test matrices with eigenvalues within a specified range. Calls to both *modcholesky* and *modchol* are demonstrated in the driver.

Appendices A and B contain the sample driver and sample output, respectively. Appendix C contains the modified Cholesky factorization code.

Note that if one wishes to process a sparse matrix A incrementally as mentioned in Section 1, the code must be simplified so that all pivoting is eliminated. In this case the calculation of Gerschgorin bound estimates is also unnecessary so the code is quite simple. The diagonal elements must still be known throughout the factorization, but the rest of the matrix can then be processed incrementally, with only the part involved in the current elimination step needed at any given iteration.

References

[1] Dennis, J. E., and Schnabel, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[2] Gill, P. E. and Murray, W. Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming 28*, (1974), 311-350.

[3] Gill, P. E., Murray, W. and Wright, M. H. *Practical Optimization*. Academic Press, London, 1981.

[4] Gould, N. Private communication, 1989.

[5] Schnabel, R. B. and Eskow, E. A New modified Cholesky factorization. University of Colorado Department of Computer Science Technical Report Number CU-CS-415-88. (To appear in *SIAM J. Sci. Stat. Computing*.)

A Sample driver

```
c
c
c       Driver for new modified cholesky factorization algorithm.
c
c

        integer n,ndim
        double precision A(100,100)
        double precision Atwo(100,100)
        double precision g(100)
        double precision maxadd
        integer pivot(100)
        double precision E(100)
        double precision eps,tau1,tau2
        integer z
        double precision high,low
        ndim=100


C       macheps subroutine computes machine epsilon,
C       the following line may be replaced by assignment to eps
C       of correct machine epsilon constant for your machine

        call macheps(eps)

C       Tolerances used by modcholesky subroutine.
C       tau1 is used in determining when to switch to phase 2 and
C       tau2 is used in determining the amount to add to the diagonal
C       of the final 2X2 submatrix.
C       The default values for these tolerances can be used
C       by calling modchol subroutine instead of modcholesky.
C       The default values for tau1 and tau2 in modchol are :  eps ** 1/3.

        tau1 = eps ** (../3.)
        tau2 = eps ** (1./3.)

C       Initial seed for random number generator used to generate test
C       matrices
        z = 1000

C       high and low are the range of the eigenvalues for the test matrix
C       to be generated.

        high = 1.0
        low = -1.0

C       The first test problem will have dimension n=4, so that the entire
C       problem can be printed out.
```

```
      n = 4

      print *,"TEST PROBLEM #1"
      print *,"Test Matrix of size",n
      print *,"with eigenvalues within the range of ",low," to ",high

      call mkmatrix(ndim,n,z,A,high,low,Atwo,g)

      print *,""
      print *,"Original 4X4 matrix"

      do 25 i=1,n
25          print (26),(A(i,j),j=1,n)
26    format (4f20.8)


      call modchol(ndim,n,A,g,eps,pivot,E)

      print *,""
      print *,"Matrix after factorization with 1 in the lower triangle"

      do 50 i=1,n
50          print (26),(A(i,j),j=1,n)

      print *,""
      print *, "Iteration  Pivot       Amt added to Aii"
      do 75 i=1,n
75          print (76),i,pivot(i),e(i)
76    format (i2,10x,i2,10x,f12.8)

      maxadd = E(n)

      print *,""
      print *,"Maximum amount added to the diagonal is",maxadd

C     The next 3 test problems have sizes n=25,50,& 75,
C     with eigenvalue ranges [-1,1],[-1,10000], & [-10000,-1] respectively.

      n = 25

      print *,""
      print *,"TEST PROBLEM #2"
      print *,"Test Matrix of size",n
      print *,"with eigenvalues in the range of ",low," to ",high


      call mkmatrix(ndim,n,z,A,high,low,Atwo,g)

      call modchol(ndim,n,A,g,eps,pivot,E)
```

```
      maxadd = E(n)
                                                                      100
      print *,"Maximum amount added to the diagonal is",maxadd


      high = 10000.0
      low = -1.0
      n = 50


      call mkmatrix(ndim,n,z,A,high,low,Atwo,g)
                                                                      110
      print *,""
      print *,"TEST PROBLEM #3"
      print *,"Test Matrix of size",n
      print *,"with eigenvalues in the range of ",low," to ",high


      call modcholesky(ndim,n,A,g,eps,tau1,tau2,pivot,E)

      maxadd = E(n)
                                                                      120
      print *,"Maximum amount added to the diagonal is",maxadd

      high = -1.0
      low = -10000.0
      n = 75

      print *,""
      print *,"TEST PROBLEM #4"
      print *,"Test Matrix of size",n
      print *,"with eigenvalues in the range of ",low," to ",high     130

      call mkmatrix(ndim,n,z,A,high,low,Atwo,g)

      call modchol(ndim,n,A,g,eps,pivot,E)

      maxadd = E(n)

      print *,"Maximum amount added to the diagonal is",maxadd


                                                                      140
      stop
      end
c*********************************************************************
c     macheps
c*********************************************************************
      subroutine macheps(eps)                                  macheps

      double precision eps
```

```
      double precision temp                                          150

      temp = 1.0

20    continue
      temp = temp / 2.0
      if ((1.0 + temp) .ne. 1.0) goto 20

      eps = temp * 2.0

      return                                                         190
      end
```

B Sample Driver Output

```
TEST PROBLEM #1
Test Matrix of size  4
with eigenvalues within the range of    -1.0000000000000 to     1.0000000000000

Original 4X4 matrix
         0.35711021         -0.10302945          0.02737268         -0.04594879
        -0.10302945          0.25254612          0.07358379         -0.38451624
         0.02737268          0.07358379          0.23396662         -0.28782367
        -0.04594879         -0.38451624         -0.28782367          0.55494709

Matrix after factorization with 1 in the lower triangle
         0.59758699         -0.10302945          0.02737268         -0.04594879
        -0.07689054          0.82587804          0.07358379         -0.38451624
         0.04580534         -0.34424172          0.49639272         -0.28782367
        -0.17240912         -0.48163633         -0.16986202          0.30827612

Iteration  Pivot      Amt added to Aii
 1           1            0.00000000
 2           4            0.13303961
 3           3            0.13303961
 4           2            0.13303961

Maximum amount added to the diagonal is   0.13303960618874

TEST PROBLEM #2
Test Matrix of size  25
with eigenvalues in the range of    -1.0000000000000 to     1.0000000000000
Maximum amount added to the diagonal is    1.2576119845957

TEST PROBLEM #3
Test Matrix of size  50
with eigenvalues in the range of    -1.0000000000000 to    10000.0000000000
Maximum amount added to the diagonal is    1.1271617927026

TEST PROBLEM #4
Test Matrix of size  75
with eigenvalues in the range of   -10000.0000000000 to    -1.0000000000000
Maximum amount added to the diagonal is    11618.452621394
```

C Modified Cholesky Factorization Code

```
C*****************************************************************
C
C       subroutine name: modcholesky
C
C       authors :  Elizabeth Eskow and Robert B. Schnabel
C
C       date    : December, 1988
C                                                                  170
C       purpose : perform a modified cholesky factorization
C                 of the form (Ptranspose)AP  + E = L(Ltranspose),
C                 where L is stored in the lower triangle of the
C                 original matrix A.
C                 The factorization has 2 phases:
C                       phase 1: Pivot on the maximum diagonal element.
C                                Check that the normal cholesky update
C                                would result in a positive diagonal
C                                at the current iteration, and
C                                if so, do the normal cholesky update.   180
C                                otherwise switch to phase 2.
C                       phase 2: Pivot on the minimum of the negatives
C                                of the lower gerschgorin bound
C                                estimates.
C                                Compute the amount to add to the
C                                pivot element and add this
C                                to the pivot element.
C                                Do the cholesky update.
C                                Update the estimates of the
C                                gerschgorin bounds.                190
C
C       input   : ndim     - largest dimension of matrix that will be used
C
C                 n        - dimension of matrix A
C
C                 A        - n*n symmetric matrix (only lower triangular
C                            portion of A, including the main diagonal, is used)
C
C                 g        - n*1 work array
C                                                                   200
C                 macheps  - machine epsilon
C
C                 tau1     - tolerance used for determining when to switch to
C                            phase 2
C
C                 tau2     - tolerance used for determining the maximum
C                            condition number of the final 2X2 submatrix.
C
C
C       output  : L      - stored in the matrix A (in lower triangular     210
```

```
C                       portion of A, including the main diagonal)
C
C               pivot - a record of how the rows and columns
C                       of the matrix were permuted while
C                       performing the decomposition
C
C               E     - n*1 array, the ith element is the
C                       amount added to the diagonal of A
C                       at the ith iteration
C                                                                      220
C
C***********************************************************************
      subroutine modcholesky(ndim,n,A,g,macheps,tau1,tau2,pivot,E)     modcholesky

      integer n,ndim
      double precision A(ndim,n),g(n),macheps,tau1,tau2
      integer pivot(n)
      double precision E(n)

C                                                                      230
C     j               - current iteration number
C     iming           - index of the row with the min. of the
C                           neg. lower Gersch. bounds
C     imaxd           - index of the row with the maximum diag.
C                           element
C     i,itemp,jpl,k   - temporary integer variables
C     delta           - amount to add to Ajj at the jth iteration
C     gamma           - the maximum diagonal element of the original
C                           matrix A.
C     normj           - the 1 norm of A(colj), rows j+1 --> n.       240
C     ming            - the minimum of the neg. lower Gersch. bounds
C     maxd            - the maximum diagonal element
C     taugamma        - tau1 * gamma
C     phase1          - logical, true if in phase1, otherwise false
C     delta1,temp,jdmin,tdmin - temporary double precision vars.
C

      integer j,iming,i,imaxd,itemp,jp1,k
      double precision delta,gamma
      double precision normj, ming,maxd                              250
      double precision delta1,temp,jdmin,tdmin,taugamma
      logical phase1

      call init(n, ndim, A, phase1, delta, pivot, g, E,
     *          ming,tau1,gamma,taugamma)

      do 10 j = 1, n-1
C
C     PHASE 1
C                                                                      260
```

```
            if ( phase1 ) then
C
C       find index of maximum diagonal element A(i,i) where i>=j
C
                maxd = A(j,j)
                imaxd = j
                do 20 i = j+1, n
                    if (maxd .lt. A(i,i)) then
                        maxd = A(i,i)
                        imaxd = i                                    270
                    end if
 20             continue

C
C       pivot to the top the row and column with the max diag
C
                if (imaxd .ne. j) then
C
C       swap row j with row of max diag
C                                                                    280
                    do 30 i = 1, j-1
                        temp = A(j,i)
                        A(j,i) = A(imaxd,i)
                        A(imaxd,i) = temp
 30                 continue
C
C       swap colj and row maxdiag between j and maxdiag
C
                    do 35 i = j+1,imaxd-1
                        temp = A(i,j)                                290
                        A(i,j) = A(imaxd,i)
                        A(imaxd,i) = temp
 35                 continue
C
C       swap column j with column of max diag
C
                    do 40 i = imaxd+1, n
                        temp = A(i,j)
                        A(i,j) = A(i,imaxd)
                        A(i,imaxd) = temp                            300
 40                 continue
C
C       swap diag elements
C
                    temp = A(j,j)
                    A(j,j) = A(imaxd,imaxd)
                    A(imaxd,imaxd) = temp
C
C       swap elements of the pivot vector
C                                                                    310
```

```
                itemp = pivot(j)
                pivot(j) = pivot(imaxd)
                pivot(imaxd) = itemp

            end if


C       Check to see whether the normal cholesky update for this
C       iteration would result in a positive diagonal.
C       and if not then switch to phase 2.                                  32

            jp1 = j+1

            if (A(j,j).gt.0) then

                do 60 i = jp1, n
                    temp = A(i,j) * A(i,j) / A(j,j)
                    tdmin = A(i,i) - temp
                    if (i .ne. jp1) then
                        jdmin = min(jdmin, tdmin)                           33
                    else
                        jdmin = tdmin
                    end if
60              continue

                if (jdmin .lt. taugamma) phase1 = .false.
            else

                phase1 = .false.
                                                                            340
            end if

            if (phase1) then
C
C       do the normal cholesky update if still in phase 1
C
                A(j,j) = dsqrt(A(j,j))
                do 70 i = j+1, n
                    A(i,j) = A(i,j) / A(j,j)
70              continue                                                    350
                do 75 i=j+1,n
                    do 80 k = j+1, i
                        A(i,k) = A(i,k) - (A(i,j) * A(k,j))
80                  continue
75              continue

                if (j .eq. n-1) A(n,n)=dsqrt(A(n,n))

            else
                                                                            360
```

```
C
C       calculate the negatives of the lower gerschgorin bounds
C
                call calcgersch(ndim,n,A,j,g)

            end if

        end if

                                                                    370
C
C       PHASE 2
C
        if (.not. phase1) then

            if (j .ne. n-1) then
C
C       find the minimum negative gershgorin bound
C
                do 90 i = j,n                                       380
                    if (i .ne. j) then
                        if (ming .gt. g(i)) then
                            ming = g(i)
                            iming = i
                        end if
                    else
                        iming = j
                         ming = g(j)
                    end if
 90             continue                                            390

C
C       pivot to the top the row and column with the
C       minimum negative gerschgorin bound
C
                if (iming .ne. j) then
C
C       swap row j with row of min gersch bound
C
                    do 100 i = 1, j-1                               400
                        temp = A(j,i)
                        A(j,i) = A(iming,i)
                        A(iming,i) = temp
 100                continue
C
C       swap colj with row iming from j to iming
C
                    do 105 i = j+1,iming-1
                        temp = A(i,j)
                        A(i,j) = A(iming,i)                         410
```

```
                        A(iming,i) = temp
105                 continue
C
C       swap column j with column of min gersch bound
C
                    do 110 i = iming+1, n
                        temp = A(i,j)
                        A(i,j) = A(i,iming)
                        A(i,iming) = temp
110                 continue                                          420
C
C       swap diagonal elements
C
                    temp = A(j,j)
                    A(j,j) = A(iming,iming)
                    A(iming,iming) = temp
C
C       swap elements of the pivot vector
C
                    itemp = pivot(j)                                  430
                    pivot(j) = pivot(iming)
                    pivot(iming) = itemp
C
C       swap elements of the negative gerschgorin bounds vector
C
                    temp = g(j)
                    g(j) = g(iming)
                    g(iming) = temp

                end if                                                440
C
C       Calculate delta and add to the diagonal.
C       delta=max{0,-A(j,j) + max{normj,taugamma},delta_previous}
C       where normj=sum of |A(i,j)|,for i=1,n.
C       delta_previous is the delta computed at the previous iteration,
C       and taugamma is tau1*gamma.
C

                normj = 0.0
                do 140 i = j+1, n                                     450
                    normj = normj + dabs(A(i,j))
140             continue

                temp = max(normj,taugamma)
                delta1 = temp - A(j,j)
                temp = 0.0
                delta1 = max(temp, delta1)
                delta = max(delta1,delta)
                E(j) =  delta
                A(j,j) = A(j,j) + E(j)                                460
```

```
C
C       update the gerschgorin bound estimates
C
                    if (A(j,j) .ne. normj) then
                        temp = (normj/A(j,j)) - 1.0

                        do 150 i = j+1, n
                            g(i) = g(i) + dabs(A(i,j)) * temp
150                         continue

                    end if
C
C       do the cholesky update
C
                    A(j,j) = dsqrt(A(j,j))
                    do 160 i = j+1, n
                        A(i,j) = A(i,j) / A(j,j)
160                 continue
                    do 165 i = j+1, n
                        do 170 k = j+1, i
                            A(i,k) = A(i,k) - (A(i,j) * A(k,j))
170                         continue
165                 continue

                else

                    call final2by2(ndim, n, A, E, j, tau2, delta,gamma)

                end if

            end if

10      continue

        return
        end
C*********************************************************************
C       subroutine name : modchol
C
C       purpose : Simple driver for the modified cholesky algorithm,
C                 with the tolerances set to the default values.
C                 i.e. tau1 = tau2 = macheps ** 1/3
C
C       input : n,ndim,A,g,macheps
C
C       output : pivot,E
C       (See subroutine modcholesky above for details on all parameters)
C*********************************************************************

        subroutine modchol(ndim,n,A,g,macheps,pivot,E)
```

modchol

```
      integer ndim, n
      double precision A(ndim,n),g(n),macheps
      integer pivot(n)
      double precision E(n)

      double precision tau1,tau2
      tau1 = macheps ** (1./3.)
      tau2 = tau1
      call modcholesky(ndim,n,A,g,macheps,tau1,tau2,pivot,E)

      return
      end
C*********************************************************************
C     subroutine name - init
C
C     purpose - set up for start of cholesky factorization
C
C     input - n, ndim, A, tau1
C
C     output - phase1    - boolean value set to true if in phase one,
C                          otherwise false.
C              delta     - amount to add to Ajj at iteration j
C              pivot,g,E - described above in modcholesky
C              ming      - the minimum negative gerschgorin bound
C              gamma     - the maximum diagonal element of A
C              taugamma  - tau1 * gamma
C
C*********************************************************************
      subroutine init(n,ndim,A,phase1,delta,pivot,g,E,ming,
     *                tau1,gamma,taugamma)

      integer n,ndim
      double precision A(ndim,n)
      logical phase1
      double precision delta,g(n),E(n)
      integer pivot(n)
      double precision ming,tau1,gamma,taugamma


      phase1 = .true.
      delta = 0.0
      ming = 0.0
      do 10 i=1,n
         pivot(i)=i
         g(i)= 0
         E(i) = 0
10    continue

c
```

```
c       find the maximum magnitude of the diagonal elements.
c       if any diagonal element is negative, then phase1 is false.
c
        gamma = dabs(A(1,1))
        if (A(1,1) .lt. 0) phase1 = .false.
        do 20 i=2,n
            if (dabs(A(i,i)) .gt. gamma) gamma=A(i,i)
            if (A(i,i) .lt. 0) phase1 = .false.
 20     continue
                                                                        570
        taugamma = tau1 * gamma

c
c       if not in phase1, then calculate the initial gerschgorin bounds
c       needed for the start of phase2.
c
        if ( .not.(phase1)) call calcgersch(ndim,n,A,1,g)

        return
        end                                                             580
C*******************************************************************************
C
C       subroutine name : calcgersch
C
C       purpose : calculate the negative of the gerschgorin bounds
C                 called once at the start of phase II.
C
C       input   : ndim, n, A, j
C
C       output  : g - an n vector containing the negatives of the       590
C                     Gerschgorin bounds.
C
C*******************************************************************************
        subroutine calcgersch(ndim, n, A, j, g)                 calcgersch

        integer ndim, n, j
        double precision A(ndim,n), g(n)

        integer i, k
        double precision offrow                                         600

        do 10 i = j, n
           offrow = 0.0
           do 20 k = j, i-1
 20           offrow = offrow + dabs(A(i,k))
           do 30 k = i+1, n
 30           offrow = offrow + dabs(A(k,i))
           g(i) = offrow - A(i,i)
 10     continue
                                                                        610
```

```
      return
      end
C*********************************************************************
C
C     subroutine name : final2by2
C
C     purpose : Handles final 2X2 submatrix in Phase II.
C               Finds eigenvalues of final 2 by 2 submatrix,
C               calculates the amount to add to the diagonal,
C               adds to the final 2 diagonal elements,
C               and does the final update.
C
C     input : ndim, n, A, E, j, tau2,
C             delta - amount added to the diagonal in the
C                     previous iteration
C
C     output : A - matrix with complete l factor in the lower trianle,
C              E - n*1 vector containing the amount added to the diagonal
C                  at each iteration,
C              delta - amount added to diagonal elements n-1 and n.
C
C*********************************************************************
      subroutine final2by2(ndim, n, A, E, j, tau2, delta,gamma)

      integer ndim, n, j
      double precision A(ndim,n), E(n), tau2, delta,gamma

      double precision t1, t2, t3,lambda1,lambda2,lambdahi,lambdalo
      double precision delta1, temp

C
C     find eigenvalues of final 2 by 2 submatrix
C
      t1 = A(n-1,n-1) + A(n,n)
      t2 = A(n-1,n-1) - A(n,n)
      t3 = dsqrt(t2*t2 + 4.0*A(n,n-1)*A(n,n-1))
      lambda1 = (t1 - t3)/2.
      lambda2 = (t1 + t3)/2.
      lambdahi = max(lambda1,lambda2)
      lambdalo = min(lambda1,lambda2)
C
C     find delta such that:
C     1.  the l2 condition number of the final
C     2X2 submatrix + delta*I <= tau2
C     2. delta >= previous delta,
C     3. lambdalo + delta >= tau2 * gamma,
C     where lambdalo is the smallest eigenvalue of the final
C     2X2 submatrix
C
```

```
      delta1=(lambdahi-lambdalo)/(1.0-tau2)
      delta1= max(delta1,gamma)
      delta1= tau2 * delta1 - lambdalo
      temp = 0.0
      delta = max(delta, temp)
      delta = max(delta1, delta)

      if (delta .gt. 0.0) then
          A(n-1,n-1) = A(n-1,n-1) + delta
          A(n,n) = A(n,n) + delta
          E(n-1) = delta
          E(n) = delta
      end if
C
C     final update
C
      A(n-1,n-1) = dsqrt(A(n-1,n-1))
      A(n,n-1) = A(n,n-1)/A(n-1,n-1)
      A(n,n) = A(n,n) - (A(n,n-1)*A(n,n-1))
      A(n,n) = dsqrt(A(n,n))

      return
      end
```

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CU-CS-443-89	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION University of Colorado 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Office
6c. ADDRESS (City, State and ZIP Code) Campus Box B-19 Boulder, CO 80309	7b. ADDRESS (City, State and ZIP Code) Post Office Box 12211 Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAL-03-88-K-0086
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Software for a New Modified Cholesky Factorization	
12. PERSONAL AUTHOR(S) Elizabeth Eskow and Robert B. Schnabel	
13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM 89/5/15 TO 90/5/15
14. DATE OF REPORT (Yr., Mo., Day) 89/08/31	15. PAGE COUNT 28
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Cholesky software, Cholesky factorization, non-positive definite

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper describes the software for a new modified Cholesky factorization recently proposed by the authors. Given a symmetric but not necessarily positive definite matrix A, the modified Cholesky factorization computes a Cholesky factorization of $A+E$, where $E=0$ if A is safely positive definite, and E is a diagonal matrix chosen to make $A+E$ positive definite otherwise. The modified Cholesky factorization was introduced by Gill and Murray and refined by Gill, Murray and Wright, and is commonly used in optimization algorithms. Our version, which is based upon new techniques, has a considerably smaller a priori upper bound on the size of E than the Gill, Murray and Wright factorization, and appears to generally produce a smaller E, and a well-conditioned $A+E$, in practice. Its cost, like the Gill, Murray and Wright version, is only a small multiple of n^2 operations greater than the standard Cholesky factorization. Thus it may be useful in optimization algorithms. We summarize our algorithm and describe the code and its use.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ OTIC USERS ☐	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Jagdish Chandra	22b. TELEPHONE NUMBER (Include Area Code) 619/549-0641	22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE